VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

MMO Hra pro programátory

MMO Game for programmers

2012 Jiří Rycka

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jiří Rycka**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: MMO hra pro programátory
MMO Game for Programmers

Zásady pro vypracování:

Navrhněte a implementujte online hru více hráčů, ve které bude hráč zastoupen automatizovanými agenty (resp. roboty). Tito agenti budou řízeni kódem, který hráč naprogramuje a nahraje do hry.
V rámci persistentního světa budou moci jednotliví agenti navzájem komunikovat. Vytvořte ukázkový svět a tutoriály pro herní API.
Implementujte uživatelské rozhraní pro hráče formou webové aplikace, které bude obsahovat:
1. Registraci a přihlášení hráčů.
2. Správu vlastních robotů s přehledem jejich stavu, výpisem historie událostí a možností nahrání zkompilovaného kódu chování robota.
3. Zobrazení stavu světa formou map.
4. Správcovské rozhraní pro ruční řízení běhu světa (pozastavení světa, nastavení parametrů robotů).

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jakub Macek**

Datum zadání: 18.11.2011
Datum odevzdání: 04.05.2012

VYSOKÁ ŠKOLA BÁŇSKÁ - TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava*.

V Ostravě 26. dubna 2012 ..... Jiří Rycka .....

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 26. dubna 2012 ..... Jiří Rycka .....

Děkuji vedoucímu bakalářské práce Ing. Jakubu Mackovi za cenné rady, připomínky a metodické vedení práce.

Abstrakt

Jedná se o návrh a implementace hry více hráčů. Hra je ovládaná uživatelskými programy skrze uživatelské API hry. Toto API je navrženo tak, aby bylo obecné a jednoduché pro pochopení. Důraz je taky kladen na výukovou funkci hry. V práci jsou popsány nejprve technologie použité pro implementaci aplikace. Následuje architektura aplikace, která je rozložená do tří částí - do webového uživatelského rozhraní, herní logiky a databáze. Dále následují pravidla hry a konkrétní technická realizace. V závěru jsou popsány aplikace podobného typu, instalace aplikace a zhodnocení celé práce.

Klíčová slova: MMO,hra, programátor, .NET, ASP.NET

Abstract

This is a design and implementation of a multi-player game. The game is controlled by user programs through the game's user API. This API is designed to be general and easy to understand. Emphasis is also placed on the educational function of the game. Technologies used for implementation of the application in this work are described first. Following is architecture of the application, which is decomposed into three parts – web user interface, game logic and database. Next are the game's rules and specific technical realization. The final part describes similar types of application, installation and evaluation of the whole work.

Keywords: MMO,game, programmer, .NET, ASP.NET

Seznam použitých zkratek a symbolů

MMO	–	Massive Multiplayer Online
LINQ	–	Language Integrated Query
SQL	–	Structured Query Language
ASP	–	Active Server Pages
HTML	–	HyperText Markup Language
XML	–	Extensible Markup Language

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

V této bakalářské práci se zabývám vývojem webové online hry pro více hráčů. Cílem tohoto projektu je implementace online hry pro více hráčů, kde veškerá přímá interakce mezi hráčem a herním světem je zastoupena programem. Každý hráč zde má k dispozici jednoho či více robotů, jejichž prostřednictvím může ovlivňovat nejen svět, ale i roboty ostatních hráčů. Hráči i roboti mohou v tomto světě mezi sebou komunikovat. Hráči posíláním zpráv, roboti díky specifickému rozhraní. Již v návrhu byl kladen důraz na rychlost a optimalizaci aplikace Při tvorbě aplikace jsem kladl důraz na to, aby aplikace byla co nejobecnější. Tzn., aby bylo možné do již hotové aplikace přidávat velmi snadno další prvky. Projevilo se to zejména při návrhu uživatelského API. Dále jsem kladl důraz na výukovou funkci. Základem je jednoduchost uživatelského API. V práci nejprve popisuji jednotlivé technologie, které jsem ve vývoji využil. V další kapitole se zabývám architekturou mnou vytvořené aplikace s klíčovými vlastnostmi a rozhodnutími, která tuto hru formovali. Následují pravidla hry a konkrétní technická realizace. Práce je zakončena ukázkou kódu robotů s důkladným popisem a srovnání s podobnými hrami.

2 Použité technologie

2.1 .NET framework

> .NET Framework je rozsáhlá softwarová platforma od společnosti Microsoft. [1]

Více informací o platformě na stránkách produktu

2.1.1 Důvody použití

Základním kritériem bylo, aby platforma nabízela možnost bezpečně oddělit vlákna uživatelů od systémových vláken hry. Je toho dosaženo za pomocí aplikačních domén .NET frameworku. Dalším důležitým aspektem byla možnost kontrolovat systémové prostředky, které dané vlákna využívají. Důležité také bylo, aby platforma poskytovala vhodné nástroje pro komunikaci s databází. Z toho důvodu je použit LINQ to SQL. A neposlední řadě také podpora webové platformy ASP.NET.

2.2 ASP.NET

> Jedná se o technologii pro tvorbu webových aplikací na platformě .NET.[2]

Hlavním důvodem pro použití je snadná integrace do .NET aplikace. Použití přineslo další výhody:

- oddělení logiky a struktury stránky
- komponentni pristup k vytvareni webovych stranek
- předpřipravené komponenty pro snadnější práci
- výrazně vyšší rychlost oproti interpretovaným jazykům.

2.3 SQL Server

Jedná se o databázový systém společnosti Microsoft.

> Tento systém nabízí celou řadu funkcí, které od moderní databáze požadujeme..[3]

Pro aplikaci využívám Edici Express, která je k dispozici zdarma a pro potřeby tohoto projektu je dostačující.

2.4 LINQ

LINQ je jeden z použitelných jazyků na platformě .NET. Díky němu je možné pracovat s kolekcemi v paměti podobně jako s databází. Je to jednoduchý a efektivní způsob pomocí dotazů, které jsou podobné jazyku SQL

> LINQ dovoluje dotazování, přidávání, filtrování, mazání a editaci dat v polích, kolekcích odvozených od generického rozhraní IEnumerable, XML či SQL databázích. [4]

V aplikaci je využíván LINQ to Object pro manipulaci s kolekcemi a LINQ to SQL pro práci s databází. Právě díky LINQ to SQL je práce s databází výrazně přehlednější.

3 Pravidla hry

Jedná se o hru robotů. Každý hráč má k dispozici jednoho či více robotů. Tito roboti se pohybují v herním světě, který odpovídá podzemí planety. Jedinou možností, jak roboty ovládat, je nahrání uživatelského kódu pro řízení robota. Kód, který bude ovládat robota po celou dobu, kdy bude robot aktivní. Jediným výstupem pro hráče je herní log, ve kterém najde veškeré chování robota od jeho spuštění. Cílem hraní hry je prohloubení znalosti algoritmického myšlení. K tomu slouží mnoho možností, které mohou hráči využít. Zejména se jedná o:

- Lokalizace surovin a mapování tunelů.
- Těžba minerálů a sběr kapalin.
- Výroba libovolných výrobků – od polotovarů k obchodování mezi hráči, přes výrobky, které lze prodávat ve světě až třeba po munici do zbraní.
- Bojovat s ostatními roboty.
- Vylepšovat robota – dokupovat nové typy vybavení či výkonnější nástroje.
- Dokupovat nové roboty a synchronizovat je s ostatními vlastními či cizími roboty.

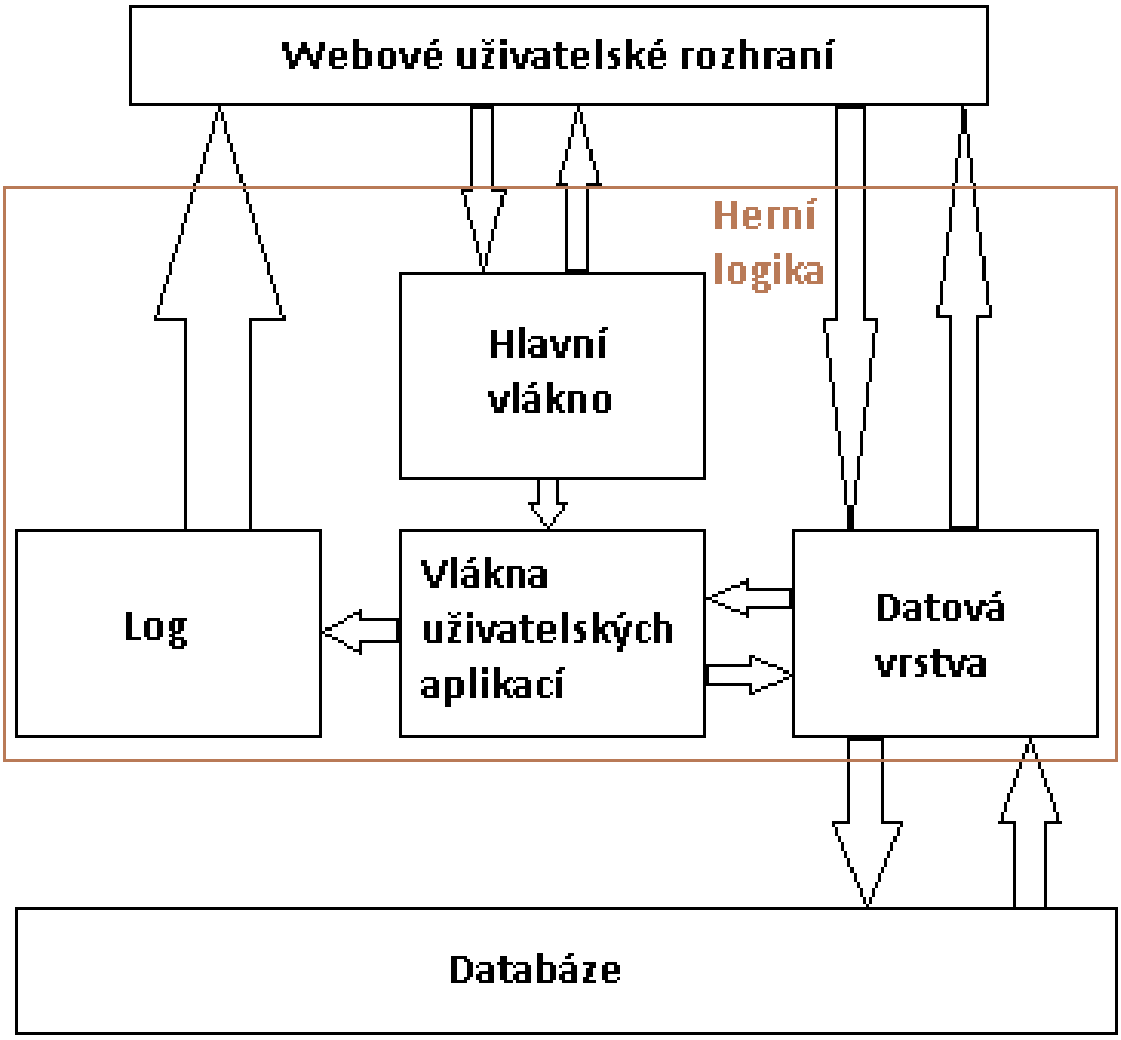
Velmi výhodným aspektem hry je fakt, že hráč nemusí být stále online. A roboti se řídí kódem po celou dobu jejich provozu. To umožňuje velmi rozsáhlé možnosti v kombinaci s výše popsanými možnostmi hry. Jednou z nejzajimavějších možností hry je možnost roboty synchronizovat tak aby byli na sobě závislí. Ve hře je toho dosaženo velmi jednoduchým způsobem. Robot může mít ve svém vybavení PublicMemoryUnit. Jedná se o několik proměnných do kterých mohou přistupovat kromě robota, který je tím vybaven, také ostatní roboti v okolí, kteří znají klíč k přístupu.

3.1 Cíle hry

Hlavním cílem hry je vytěžení speciální rudy umístěné hluboko ve světě. Aby hráč vyhrál musí být veškerá ruda z herní mapy vytěžena. Až se tak stane, vyhrává hráč s nejvyšším počtem jednotek této rudy. Rudu lze získat těžbou či obchodováním s ostatními hráči, lze ji také pomocí výrobních zařízení extrahovat z jiných rud, což je ovšem velmi časově náročné.

4 Architektura

4.1 Diagram architektury



Obrázek 1: Diagram architektury

4.2 Herní logika

Herní logika je kompletně oddělená od ostatních částí hry. Je napsána v jazyce C#. Je uložena ve jmenném prostoru BaseGame. Jedná se o multithreadovou aplikaci. Jádro tvoří několik základních vláken, která se starají o chod světa a podporují některé specifické

vybavení. Kódy robotů, které nahrají do hry hráči, běží v dalších oddělených vláknech a v oddělených aplikačních doménách.

4.2.1 MainThread

Hlavní třída celé herní logiky. Jejím jediným cílem je aktivace herního jádra a práce s vlákny pro uživatelem nahrané kódy. Mimojiné obsahuje kolekci všech robotů, kteří jsou aktuálně aktivní – tzn. jejich kód je spuštěn. Tato třída vznikla protože bylo nutné uchovávat veškeré aktuálně spuštěné roboty a také všechny vytvořené aplikační domény.

4.2.2 Aplikační domény

Bez použití aplikačích domén by každá chyba v hráčském kódu mohla způsobit pád celé aplikace. Proto každý spuštěný robot má svou vlastní aplikační doménu. V ní je spuštěn hlavní kód robota. Hlavním důvodem pro použití aplikačních domén ochrana právě před nebezpečnými kódy. Nyní když uživatel nahraje škodlivý kód, způsobí to maximálně ukončení a uvolnění aplikační domény, kterou má jeho robot k dispozici. Dalším důvodem proč použít aplikační domény je optimalizace. Díky aplikačním doménám je totiž možné sledovat, kolik dané vlákno využívá systémových prostředků. Pokud tyto prostředky přesáhnou kritickou hranici je možné toto vlákno ukončit a aplikační doménu uvolnit.

4.2.3 Herní timer

Velkým problém návrhu aplikace bylo, aby všichni hráči měli stejné podmínky pro hraní. A to zejména, aby všechny metody API trvaly stejnou dobu. Dalším problémem byla velká náročnost na výpočetní výkon, pokud by nebyla žádná optimalizace v herním API.

Proto byl vytvořen herní timer. Tento timer měří herní čas a vyvolává tzv. Tiky (s časovou periodou 10 ms). Všechny metody, které může hráč využívat, mají určitou časovou náročnost – respektive mají explicitně určeno počet tiků, během kterých se budou vykonávat. Tento čas je mnohonásobně vyšší než čas, který metoda standardně potřebuje ke svému dokončení, v některých případech i v řádu několika sekund. Toto zpoždění optimalizuje API a také dává udává stejné podmínky pro všechy hráče. Je to díky tomu, že v době mezi standardním koncem metody a koncem definovaným časem je vlákno, které řeší daný kód, uspáno.

Prakticky je to řešeno statickou třídou Core. Tato třída obsahuje kolekci obsahující reference na vlákna a hodnotu tiku, při kterém má být dané vlákno probuzeno. Dále tato třída obsahuje samotný Timer, který každý tik vyvolá událost, která probudí všechny vlákna která, mají hodnotu tiku menší nebo stejnou jako je aktuální tik. Všechny metody, které vyžadují časovou náročnost, mají stejnou strukturu. Každá z nich končí metodou, která toto vlákno dočasně uspí.

Pomocí herního timeru je taktéž realizováno kompletní zastavení světa a opětovné spuštění. Je to realizováno pomocí proměnné typu bool. Pokud je tato proměnná nastavená na false, třída Core neprobouzí uspaná vlákna a dojde k postupnému zastavení světa. Pokud se hodnota změní na true, svět se postupně spustí.

4.2.4 Připojení uživatelského kódu

Původní návrh byl vytvářet pro každé konkrétní vybavení zvlášť třídu, která by jej obsluhovala. Problémem tohoto návrhu aplikace bylo, jak přidat a spustit hráčem napsaný kód, aby hráč neměl možnost jakkoli podvádět a využívat všechny metody ve hře, ale jen ty které jsou pro něj určené. Ale aby zároveň měl přístup ke statiskám svého robota.

Proto byl návrh v tomto ohledu změněn. Nyní aplikace obsahuje jen třídy pro každou kategorii vybavení (např. pohon, pancíř...). Tyto třídy obsahují veškeré metody pro práci s daným vybavením a to jak ty pro hráče, tak ty které jsou pro řešení v rámci systému. Aby se tyto dva typy metod rozlišily, každá třída implementuje rozhraní, které obsahuje pouze metody, ke kterým má přístup hráč. Pokud hráč pak ve svém kódu získá instanci nějakého vybavení tak ta instance je přetypovaná na toto rozhraní.

Samotný uživatelský kód pak není přímo spojen s datovou strukturou robotů. Každý robot má definováno několik základních událostí. Právě na tyto událostí je připojen kód robota, který do hry nahrají hráči.

Dalším důvodem proč tento návrh vznikl, byla snaha mít možnost v budoucnu přidat do hry další události, na které by mohli hráči reagovat například poškození robota, bez nutnosti měnit tento návrh.

Prakticky je to řešeno velmi snadno. Hráčské metody jsou připojeny na události objektů typu Robot. Při spuštění robota je spuštěna událost MainScript, která se řídí hlavním kódem robota. Tento uživatelem definovaný kód využívá metody z herního API. Každá metoda má kromě výše zmíněné práce s Timerem kontrolní metodu, která v případě potřeby spouští další události. Tzn. Každá metoda obsahuje tento kod:

```
robot.SolveEvents();
// TĚLO METODY
Core.Core.SleepAndWake();
//return – návrat z funkce
```

Výpis 1: Struktura každé herní metody

4.2.5 Herní log

Velmi důležitým aspektem hry je odezva robota. Hráč musí mít k dispozici nějaký nástroj, pomocí ktetého zjistí zda kód, který napsal, funguje přesně jak hráč zamýšlel. Proto vznikl herní log, který je kromě statistiky robotů pro hráče jediným dalším prostředkem ke zjištění co přesně robot dělá. Každá metoda uživatelského API zapisuje svůj výsledek do logu robota. Tento zápis je řešený pomocí vláken z třídy ThreadPool, takže nikterak nezpomaluje metodu, která do logu svůj stav zapisuje. Kromě této vlastnosti má hráč také možnost sám do Logu zapisovat. Hráč si může log daného dne stáhnout přes uživatelské rozhraní.

4.2.6 Datová vrstva

Pro komunikaci s databází je určená standardní LINQ to SQL třída. LINQ to SQL je velmi silným nástrojem pro práci s databází. Umožňuje pracovat s entitami databáze stejně,

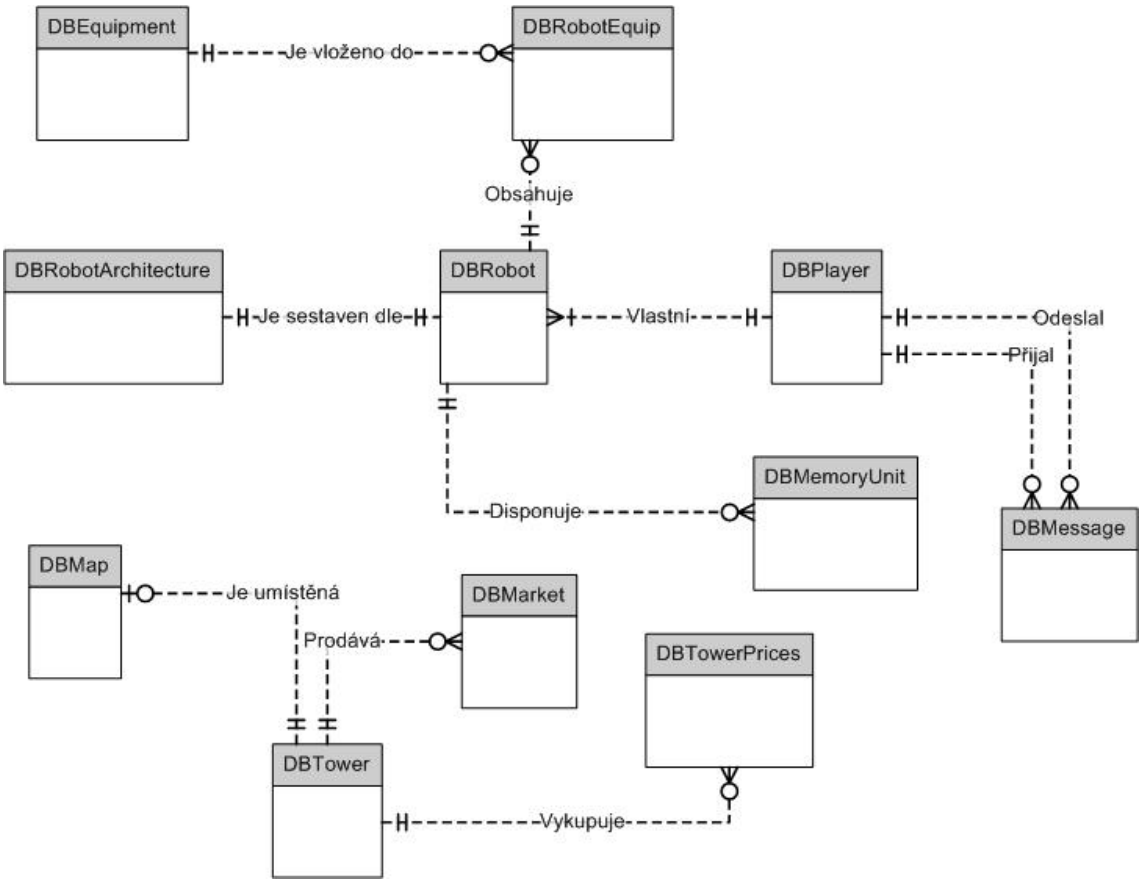
jako by to byly kolekce. Díky tomu je komunikace s databází přehlednější, jednodušší a také vice objektově zaměřená.

4.2.7 Návrh uživatelského API

API bylo navrženo tak, aby umožňovalo více stejného nebo podobného vybavení na jednom robotu. Základem API je rozhraní IRobot, jehož metody poskytují přístup k instancím dalších zařízení a výbavy. Dále obsahuje metody pro získání stavu některých hlavních parametrů robota. Více v kapitole Technická realizace.

4.3 Databáze

Hlavním datovým úložištěm aplikace je její databáze. Jedná se o SQL Server.



Obrázek 2: Diagram databáze

4.3.1 Datový slovník

Tabulky s datovým slovníkem umístěny v přílohách.

4.4 Uživatelské rozhraní

Součástí aplikace je webové uživatelské rozhraní realizované pomocí technologie ASP.NET. Uživatelské rozhraní se skládá ze dvou částí. Části administrátorské a části uživatelské.

4.4.1 Administrátorská část

Velmi jednoduché administrátorské rozhraní, zobrazující několik základních voleb pro ovlivňování světa. A to volbu pro zastavení a opětovné spuštění herního světa. Dále pak poskytuje rozhraní pro sledování robotů od uživatelů v určité oblasti herního světa s možností stáhnout jejich logy.

4.4.2 Uživatelská část

Uživatelská část nabízí několik možností. Mezi ně patří:

- Registrace a přihlášení hráče.
- Správa herního profilu.
- Komunikace mezi hráči. Možnost posílat zprávy jiným hráčům a číst příchozí zprávy.
- Zobrazení stavu svých robotů v jednoduchém seznamu spolu se základními atributy.
- Možnost zobrazit detail robota, který obsahuje rozhraní pro nahrání knihovny pro jeho řízení, spuštění robota,násilné zastavení robota a stáhnutí herního logu daného robota.
- Možnost pro zakoupení nového robota - vybrání architektury. (robot získá základní komunikátor pro nakoupení zbylého vybavení)

4.4.2.1 Registrace hráče Při registraci musí uživatel zadat uživatelské jméno, heslo a email. Pokud je email a uživatelské jméno neobsazeno je registrace úspěšná. Po úspěšné registraci se hráči vytvoří profil a získá první robota. Robot je vybaven základním vybavením. Poté již hráč může nahrávat svůj kód robotovi. Taktéž je robotovi náhodně vybraná jeho bázová věž a tím i jeho počáteční souřadnice.

4.4.2.2 Spuštění hráčského kódu Spuštění kódu je možné poté, co hráč nahraje kód přes uživatelské rozhraní. Po spuštění kódu dojde k vytvoření aplikační domény. V této doméně je pak vytvořena instance třídy z knihovny kterou hráč nahraje do hry a vytvoření instance základní třídy Robot. Metody z hráčské knihovny jsou následně přidány na události základní třídy Robot. Následuje vytvoření nového vlákna a jeho spuštění. V rámci tohoto procesu jsou všechny důležitá data uložena do kolekcí.

4.4.2.3 Násilné ukončení hráčského kódu K násilnému ukončení může dojít ve třech případech:

- pokud hráč v rámci svého uživatelského rozhraní ukončí činnost robota
- pokud administrátor v rámci svého uživatelského rozhraní ukončí činnost robota
- pokud vlákno kontrolující využité systémové prostředky aplikačních domén zjistí, že hráčův kód spotřebovává příliš systémových zdrojů

Průběh tohoto ukončení je pak velmi jednoduchý. Nejprve je ukončeno vlákno daného robota. Poté je do Logu robota přidána informace o ukončení činnosti robota. A končí to uvolněním dané aplikační domény.

5 Technická realizace

Popis vybraných částí aplikace.

5.1 Perzistentní svět

Svět je reprezentován herní mapou, která má tři rozměry. Základem je standardní čtvercová síť o dvou rozměrech (x a y), kterou doplňuje třetí rozměr Floor. Tento rozměr určuje patro herního světa. Nejmenším prvkem herní mapy je tedy jedno pole o třech souřadnicích, které se nazývá buňka. Každá buňka má několik parametrů. Hlavním parametrem, který určuje její chování a způsoby manipulace s ní, je typ buňky. Každá buňka musí mít definovaný typ. Mezi typy patří:

- BLOCK – neprůchozí buňka
- FIXED_TUNNEL – průchozí buňka pro všechny typy robotů
- UNFIXED_TUNNEL – buňka s omezenou průchodností, lze fixovat
- LIQUID – buňka s omezenou průchodností, obsahuje kapalinu, kterou lze sbírat. Po vysbírání se z ní stává HOLE.
- LAVA – buňka s omezenou průchodností, na této buňce může být robot zničen
- MINERAL – neprůchozí buňka, obsahuje minerál, který lze vytěžit. Po vytěžení se z ní stává UNFIXED TUNNEL.
- HOLE - buňka s omezenou průchodností, když je zasypána stává se z ní UNFIXED TUNNEL.
- BRIDGE – specifická buňka, slouží k přechodu mezi patry herního světa.
- TOWER – buňka obsahující věž, jejímž prostřednictvím komunikují roboti s herním světem.

Věže dělí na 5 typů:

- BASE_TOWER - bázová věž. Každý hráč má jednu, první robot začíná právě v této věži.
- REPAIR_TOWER – věže, sloužící k opravování robotů
- MATERIAL_TOWER – věže vykupující těžitelné suroviny,polotovary a výrobky
- TRANSFER_TOWER – věže přesunující suroviny mezi roboty.
- MARKET_TOWER - věže prodávající vybavení pro roboty.

Tyto typy se liší jak přístupem na dané pole, jelikož roboti musí mít potřebné vybavení pro pohyb po specifickém typu pole a pak také možnostmi, které daný typ pole nabízí. Každý z těchto typů má několik dalších parametrů.

5.2 Roboti

Roboti jsou programovatelné jednotky, které řídí hráčem definovaný kód. Hráč u robota definuje jeho hlavní kód, který bude řešit veškeré chování robota. Dále může hráč implementovat metody pro události, které mohou nastat mimo hlavní kód. Například poškození robota. Každý robot je definován svými hlavními parametry, architekturou a vybavením, kterým disponuje. Hlavní parametry robota jsou:

- Souřadnice na herní mapě (x,y,Floor)
- Souřadnice základní věže na herní mapě (x,y,Floor)
- Aktuální počet zásahových bodů (HitPoints)
- Směr robota při pohybu vpřed (Direction)

Tohle je základní interface pro robota:

```
interface IRobot
{
    int X{ get; }
    int Y{ get; }
    int Floor{ get; }
    int HitPoints{ get; }
    int Direction{get;}
    int BaseX { get; }
    int BaseY { get; }
    int BaseFloor { get; }
    string Key{set; get; }
    int GetID();
    EqPower LoadPowerBySlot(int slot);
    EqArmor LoadArmorBySlot(int slot);
    EqStorage LoadStorageBySlot(int slot);
    EqWeaponStandard LoadStandardWeaponBySlot(int slot);
    EqTeleportationUnit LoadTeleportationUnitBySlot(int slot);
    EqCommunicationUnit LoadCommunicationUnitBySlot(int slot);
    EqFixingTunnel LoadTunnelFixingUnitBySlot(int slot);
    EqGatheringLiquid LoadLiquidGathererBySlot(int slot);
    EqHloubkomer LoadHloubkomerBySlot(int slot);
    EqMineralDetector LoadmineralDetectorBySlot(int slot);
    EqMining LoadMiningUnitBySlot(int slot);
    EqPrivateMemoryUnit LoadPrivateMemoryUnitBySlot(int slot);
    EqPublicMemoryUnit LoadPublicMemoryUnitBySlot(int slot);
    EqRadar LoadRadarBySlot(int slot);
    void Wait(int milisecond);
    void Log(string message);
    int Random(int max);

}
```

Výpis 2: Standardní rozhraní IRobot

5.3 Architektura robota (RobotArchitecture)

Každý robot má definovanou svou architekturu. Architektura definuje zejména, jaké typy vybavení může robot využívat. Každá architektura definuje základ zásahových bodů robota a základní cenu za opravu jednoho zásahového bodu a v neposlední řadě také maximální patro, do jakého se lze s danou architekturou dostat. Hlavním parametrem architektury robota je ale rozložení slotu pro vybavení. Každý robot může být vybaven až deseti různými zařízeními. Tato zařízení se zasazují do slotů v architektuře robota. Slotů je několik typů:

- A Slot – Slot pro podvozek, každý robot má zpravidla maximálně jeden takový slot.
- B Slot – Slot po celém obvodu robota, využívá se zejména na pancíř, ale není to pravidlo.
- C Slot – Otevíratelný slot, zejména využívaný pro nákladový prostor.
- D Slot – Hlavní slot – umístěný v horní části robota. Jedná se o většinu aktivních zařízení.
- E Slot - Malý slot pro vnější vybavení, zejména senzory, umístěný na slotu B nebo uvnitř něj.
- F Slot - Slot pro vnitřní zařízení.

Do slotů lze vybavit různé typy vybavení. Není zde žádná vazba mezi typem vybavení a slotem, resp. různé vybavení jednoho typu mohou být v různých slotech.

5.3.1 ProjectileManager

Na rozdíl od robotů, jejichž veškeré akce probíhají sekvenčně dle jejich hlavního kódu, jsou zde další objekty, jejichž pohyb a interakci je nutno řešit odděleně. Jedná se o projektily zbraní, které roboti používají. Tato třída vznikla v druhé verzi hry. Původně byla střelba řešená přímo uvnitř metod zbraní. Vždy tedy byl po výstřelu ze zbraně řešen let projektilu a jeho následný dopad se všemi efekty pro okolí. Toto řešení nebylo příliš vhodné, protože při větším dosahu zbraní byla tato akce velmi časově náročná. Proto vznikla tato třída. Její fungování je podobné fungování herního Timeru. Při výstřelu metoda zbraně registruje projektil u ProjectileManageru, který jej začne řešit. Prakticky to funguje tak že registrace jen přidá instanci projektilu do kolekce této třídy. ProjectileManager kontinuálně kontroluje tuto kolekci a pokud je do této kolekce přidán nový záznam, spustí automaticky řešení tohoto záznamu.

5.4 Vybavení robotů (Equipments)

Vybavení robotů je hlavní aspekt hry z pohledu hráče. Robot sám o sobě poskytuje hráči jen minimální možnosti a to zejména informativní nebo možnosti týkající se právě správy vybavení. Je několik typů vybavení. Každý typ vybavení má vlastní parametry a metody. Proberu všechny typy vybavení spolu s rozhraním, které nabízejí.

5.4.1 Armor

Slouží k obraně robota, disponuje redukcí poškození proti určitému typu zbraně a může přidávat robotovi bonus do zásahových bodů.

```
public interface IEquipmentArmor
{
    int GetArmorReduction();
    int GetArmorType();
    int GetArmorHPBonus();
}
```

Výpis 3: Rozhraní IArmor

5.4.2 CommunicationUnit

Díky tomuto zařízení lze komunikovat s roboty (pomocí veřejné paměťové jednotky) a komunikovat s věžemi v herním světě.

```
public interface IEquipmentCommunicationUnit
{
    int GetPricePerHitPointFromRepairTower(int towerX, int towerY, int hpToRepair);
    void SendRequestMessageToRepairTower(int towerX, int towerY, int hpToRepair);
    int GetPricePerUnitFromMaterialTower(int towerX, int towerY, int itemType);
    void SendRequestToMaterialTower(int towerX, int towerY, int iSlotID);
    int GetInfoFromMarketTower(int towerX,int towerY,int equipmentID);
    void SendRequestToMarketTower(int towerX, int towerY, int iEquipID, int iSlotID);
    void SetMemoryUnit(string sKey, int iValueNumber, string sValue);
    void SendRequestToTransferTower(int storage,int count, string password);

}
```

Výpis 4: Rozhraní ICommunicationUnit

5.4.3 CraftingUnit

Slouží k výrobě produktu z jednoho až tří surovin.

```
public interface IEquipmentCraftingUnit
{
    void DoCraft(int iSlotWithMaterial1, int iSlotWithMaterial2, int iSlotWithMaterial3, int
        iSlotForProduct);
    int GetTimeToWork();
}
```

Výpis 5: Rozhraní ICraftingUnit

5.4.4 FixingUnit

Díky tomuto zařízení lze zpevnit buňku herní mapy z UNFIXED na FIXED a umožnit tak přístup všem robotům.

```
public interface IEquipmentFixingTunnel
{
    int GetTimeToWork();
    int GetPercentOfFixingProcess();
    void Work();
}
```

Výpis 6: Rozhraní IFixingUnit

5.4.5 GatheringLiquid

Zařízení pro sběr kapalin určitého typu.

```
public interface IEquipmentGatheringLiquid
{
    int GetTimeToWork();
    int GetGatheringLiquidItemType();
    void Work();
    void SetGatheringLiquidInventory(int slot);
    int GetGatheringLiquidInventory();
}
```

Výpis 7: Rozhraní IGatheringLiquid

5.4.6 MineralDetector

Zařízení pro zjištění typu minerálu nebo kapaliny.

```
public interface IEquipmentMineralDetector
{
    int GetTimeToWork();
    int GetMineralType();
}
```

Výpis 8: Rozhraní IMineralDetector

5.4.7 Mining

Slouží k těžbě minerálu.

```
public interface IEquipmentMining
{
    int GetTimeToWork();
    int GetMiningItemType();
    void Work();
    void SetMiningInventory(int slot);
    int GetMiningInventory();
}
```

Výpis 9: Rozhraní IMining

5.4.8 Power

Slouží k pohybu robota, obsahuje pohon a podvozek a definuje, na jaký typ buňky herní mapy se může robot dostat.

```
public interface IEquipmentPower
{
    int MoveForward();
    int TurnLeft();
    int TurnRight();
    int GetMovementDelay();
    int GetTurningDelay();
}
```

Výpis 10: Rozhraní IPower

5.4.9 PrivateMemoryUnit a PublicMemoryUnit

Slouží k uschování dat při běhu programu. Data, která nejsou v této jednotce uložena, mohou být ztracena. Do private jednotky má přístup pouze kód robota, který má tuto jednotku vybaven. Public jednotka má stejné vlastnosti jako priváte, navíc ale může cizí robot zapisovat data do této jednotky, pokud zná klíč přístupu.

```
public interface IEquipmentMemoryUnit
{
    void SetData(int valueNumber,string newValue);
    string GetData(int valueNumber);
}
```

Výpis 11: Rozhraní IMemoryUnit

5.4.10 Radar

Slouží k identifikaci buňky, mapy či umístění robotů.

```
public interface IEquipmentRadar
{
    int GetMapCellType();
    int GetMapCellType(Coordinates coordinates, int searchForRobot);
    Coordinates GetNearestMapCellTypeExpect(int iMapCellType, Coordinates cExcept);
}
```

Výpis 12: Rozhraní IRadar

5.4.11 Storage

Nákladový prostor – veškerá zařízení, která pracují s nějakými předměty, musí mít nákladový prostor – např. zbraně musí odněkud brát munici.

```
public interface IEquipmentStorage
{
    int GetInventoryType();
    int GetInventorySize();
    int GetInventoryFilling () ;
}
```

Výpis 13: Rozhraní IStorage

5.4.12 SaveUnit

Jednotka pro první roboty každého hráče, má vylepšenou funkci, dokáže zajistit přenesení robota do bázové věže.

```
public interface ISaveUnit
{
    void BackToBase();
}
```

Výpis 14: Rozhraní ISaveUnit

5.4.13 Standard Weapon

Standardní projektilová zbraň.

```
public interface IEquipmentWeaponStandard
{
    int GetRangeMin();
    int GetRangeMax();
    int GetTimeToFire();
    int GetAmmoType();
    void SetRange(int range);
    void Fire() ;
    void SetAmmoInventory(int slot);
    int GetAmmoInventory();
}
```

Výpis 15: Rozhraní IStandardWeapon

6 Příklad pro vysvětlení API

Jako příklad uvádím tři roboty. První dva také demonstrují komunikaci mezi roboty.

6.1 Robot 1

Tento robot těží rudu z nerostu. A vyčkává na signál od druhého robota. Pokud dostane signál o nebezpečí nebo naplní celý nákladovy prostor rudou, přenese se do své bázové věže pomocí teleportace.

```
public override void MainScript(IRobot robot)
{
    /*konstanta pro paměťovou buňku*/
    int BEZPECNO = 1;
    /*Definuje přístupovy klíč, který je nutný pro komunikaci s tímto robotem. Pouze roboti kteří
        znají tento klíč mohou s tímto robotem komunikovat.*/
    robot.Key = "Pristup";
    /*Získání instance vybavení pro veřejnou pamětovou jednotku*/
    IEquipmentMemoryUnit publicUnit = robot.LoadPublicMemoryUnitBySlot(3);
    /*Nastavení proměnné do veřejné paměťové jednotky*/
    publicUnit.SetData(BEZPECNO,"ok");
    /*Získání instance těžebního nástroje*/
    IEquipmentMining mining = robot.LoadMiningUnitBySlot(2);
    /*Nastavení do jakého úložného prostoru má těžební nástroj přesouvat vytěženou rudu*/
    mining.SetMiningInventory(4);
    /*Získání instance úložiště*/
    IEquipmentStorage storage = robot.LoadStorageBySlot(4);
    *Získání instance detektoru minerálů*/
    IEquipmentMineralDetector detector = robot.LoadMineralDetectorBySlot(7);
    /*Získání instance teleportační jednotky*/
    IEquipmentTeleportationUnit teleport = robot.LoadTeleportationUnitBySlot(8);
    /*Cyklus s kontrolou zda mineral před robotem je železo*/
    while (detector.GetMineralType() == ItemTypeConstants.IRON)
    {
        /*Kontrola zda se nezměnil obsah veřejné paměťové jednotky nebo zda naplnění úložiště
            není kompletní*/
        if ((!publicUnit.GetData(BEZPECNO).Equals("ok")) || storage.GetInventoryFilling() ==
            storage.GetInventorySize())
        {
            /*Pokud ano, teleporuje robota do bázové věže*/
            teleport.BackToBase();
        }
        else
        {
            /*Robot vytěží jednu jednotku minerálu a cyklus se opakuje*/
            mining.Work();
        }
    }
}
```

Výpis 16: Kód prvního robota

6.1.1 Vysvětlení kódu

Základem je získání instancí všech vybavení, které chceme využívat. Například pro těžební nástroj je to:

IEquipmentMining mining = robot.LoadMiningUnitBySlot(2);.

Pomocí tohoto příkazu získáme do rozhraní určeného pro práci s daným vybavením instanci daného vybavení. Na tento objekt pak můžeme používat metody, které dané rozhraní poskytuje. Tento robot těží rudu z nerostu. A vyčkává na signál od druhého robota. Pokud dostane signál o nebezpečí nebo naplni cely nakladovy prostor rudou přenese se do své bázové věže pomocí tele

6.2 Robot 2

Tento robot hlídá přístupovou cestu. Pokud se objeví jiný robot, pošle signál robotu u rudy.

```
public override void MainScript(IRobot robot)
{

    int BEZPECNO = 1;
    /*Nastaví si stejný přístupový klíč jako Robot 1*/
    string key = "Pristup";
    /*Získá instance potřebného vybavení*/
    IEquipmentRadar radar = robot.LoadRadarBySlot(3);
    IEquipmentCommunicationUnit com = robot.LoadCommunicationUnitBySlot(5);
    /*Pracuje v nekonečném cyklu*/
    while (true)
    {
        /*Kontroluje pole s přístovou cestou, a hledá na ni roboty*/
        if (radar.GetMapCellType(new Coordinates(55,75,1),searchForRobot))
        {
            /*Pokud najde robota, vyšle do prostoru signál který všem robotům, kteří mají jako klíč
                nastavený obsah proměnné key a mají veřejnou paměťovou jednotku, změní
                hodnotu proměnné 1 na "robot" */
            com.SetMemoryUnit(key, BEZPECNO, "robot");
        }
        /*Počká 500 milisekund a celou akci opakuje*/
        robot.Wait(500);
    }
}
```

Výpis 17: Kód druhého robota

6.3 Robot 3

Ukázka robota, který funguje samovolně - hráč se o něj již nemusí starat. Robot jezdí po mapě, hledá minerály, kapaliny a jejich souřadnice a typy ukládá do logu.

```
public override void MainScript(IRobot robot)
{
```

```
/*Získání instance detektoru minerálů*/
IEquipmentMineralDetector detector = robot.LoadMineralDetectorBySlot(7);
/*Nastavení proměnné do veřejné pohon*/
IEquipmentPower power = robot.LoadPowerUnitBySlot(1);
/*Získá instance radaru*/
IEquipmentRadar radar = robot.LoadRadarBySlot(3);
while (true)
{
    int random;
    /* Zjisti typ pole*/
    int typ_pole = radar.GetMapCellType();
    /*Zkontroluje zda je typ mineral nebo kapalina*/
    if (typ_pole == MapConstants.MAP_CELL_MINERAL || typ_pole == MapConstants.
        MAP_CELL_LIQUID)
    {
        /*Pokud ano, zjisti konkrétní typ*/
        int subtype = detector.GetMineralType();
        /*Uloží souřadnice, smět natočení a typ suroviny do logu*/
        robot.Log(robot.X.ToString() + "," + robot.Y.ToString() + robot.Floor.ToString() +robot.
            Direction.ToString() +":" + subtype.ToString());
        /*Vygeneruje náhodné číslo 1 nebo 2*/
        random = robot.Random(2);
        /*Otočí se náhodně,bud doleva nebo doprava (rovně je minerál nebo kapalina a přes ty
            není možný přesun) */
        if (random == 1)
        {
            power.TurnLeft();
        }
        else
        {
            power.TurnRight();
        }
    }
    else
    {
        /*Zkontroluje zda se jedná o tunel, kam může vjet*/
        if (typ_pole == MapConstants.MAP_CELL_FIXED_TUNNEL)
        {
            /*Pokud ano, posune se o jedno pole vpřed*/
            power.MoveForward();
        }
        else
        {
            /*Pokud pole není ani surovina ani opravený tunnel, jedná se o nebezpečné pole, proto
                opět změní náhodně směr*/
            random = robot.Random(2);
            if (random == 1)
            {
                power.TurnLeft();
            }
            else
            {
                power.TurnRight();
            }
```

```
                //a cyklus se znova opakuje
            }
        }
    }
}
```

Výpis 18: Kód třetího robota

7 Popis instalace

7.1 Instalace na straně serveru

Postup:

1. Mít funkční IIS.
2. Nainstalovat MS SQL Server.
3. Připravit novou aplikaci v IIS.
4. Nahrát do IIS složku s hrou.
5. Vytvořit vlastní databazi a naplnit ji. Nebo využít SQL skript, který malý ukázkový svět vytvoří.
6. Do konfiguračního souboru settings.xml ve složce hry zadat základní nastavení:
   - ConnectionString na databázi.
   - Cestu pro ukládání pomocných souborů.
7. Spuštění aplikace.

7.2 Připojení do hry na straně hráče

Postup:

1. Mít vhodné vývojové prostředí s možností kompilace .NET aplikací.
2. Založit si nový projekt.
3. Přidat do něj knihovu UserBaseGame.
4. Napsat kód pro ovládání robota dle šablony.
5. Zkompilovat kód.
6. Přihlásit se do hry přes uživatelské rozhraní.
7. Vybrat konkrétního robota.
8. Nahrát knihovnu.
9. Spustit robota.
10. (Stáhnout si log robota.)

8 Srovnání s podobnými hrami

8.1 Robot Karel

Vyzkoušel jsem si programování v Karlovi (Jazyku pojmenovaném po spisovateli Karlu Čapkovi), návodů a informací z výuky metodiky programování Standfordské university. Tento kurz je možné virtuálně navštívit online online a také získat informace na stránkách university. Jedná se o velmi jednoduchou hru pro výuku programování, kde hráči mají k dispozici 5 metod, pomocí kterých mohou řešit úlohy. Hlavním výhodou je tedy jednoduchost. Proto je vhodný jen pro výuku naprostých základů.

```
BEGINNING-OF-PROGRAM

 DEFINE turnright AS
 BEGIN
    turnleft
    turnleft
    turnleft
 END

 BEGINNING-OF-EXECUTION
   ITERATE 3 TIMES
      turnright

    turnoff
END-OF-PROGRAM
```

Výpis 19: Ukázka programu v Karlovi

8.2 Colobot

Jedná se o hru od společnosti EPSITEC. Základem je kolonizování planety pomocí robotů, kterým může naprogramovat chování. Jedná se o 3D hru pro jednoho hráče. Hra obsahuje několik herních možností:

- Missions – část čistě pro zábavu, hráč plní se svými roboty a omezenými surovinami určitě mise. Během těchto misí hráč není nucen nijak využívat programování.
- Exercise – čistě výuková část, hráč má zadaný konkrétní úkol, kterého musí dosáhnout za pomocí naprogramování robota. K úkolu je velmi propracovaná nápověda.
- Challenge – podobná část jako Exercise, ovšem hráč už nemá k dispozici žádnou nápovědu a musí problém vyřešit vlastními znalostmi.
- Free Game - poslední část, ve které hráči mohou sami dle vlastních nápadů realizovat složitější problémy.

Jedná se o hru, na které pracoval větší tým vývojářů. Výhodnou této hry pak je 3D grafika, rozsáhlá dokumentace (v angličtině). Nevýhodou je v tomto ohledu příliš velká jednoduchost. A také cena aplikace.

```
extern void object::Zabij()
{
    object item;
    aim(-20);
    while (true)
    {
        while (radar(AlienAnt, 0, 360, 0, 20) == null)
        {
            item = radar(AlienAnt);
            turn(direction(item.position));
            motor(1,1);
            jet(0);
            if (position.z-topo(position) < 6)
            {
                jet(1);
            }
            if (position.z-topo(position) > 9)
            {
                jet(-1);
            }
            wait(0.2);
        }
        fire(1);
    }
}
```

Výpis 20: Ukázka programu v Colobot

8.3 Srovnání s tímto projektem

Mezi jednoznačné výhody patří možnost hry více hráčů. Dále pak použití webového rozhraní. To znamená, že do hry se připojíte z jakéhokoli počítače. Další výhodou je velmi intuitivní programování využívající objekty a rozhraní. Dále pak je hra definuje pouze principy a chování jednotlivých typů nástrojů. Samotný konkrétní herní obsah je libovolně rozšiřitelný. Výuková část je srovnatelná s hrami, popsanými výše. Nevýhodou ovšem je absence grafiky a rozsáhlejší dokumentace.

9 Závěr

Cílem této práce bylo vytvořit hru pro více hráčů, kde akce hráče jsou zastoupeny programem. Tento cíl jsem splnil. Aplikace je funkční a obsahuje mnoho herních možností. Na počátku vývoje této aplikace byly mé znalosti dané problematiky velmi omezené a kusé. Neznal jsem vůbec platformu .NET. Proto i návrh celé aplikace nese tuto skutečnost.Nicméně celý návrh a implementace mě obohatili o spostu nových znalostí problematiky, mezi které patří práce s vlákny, aplikačními doménami, platformou ASP.NET a mnoho dalších znalostí. Slabší částí aplikace je tedy uživatelské rozhraní, které není vhodně navrhnuté a bylo by vhodné jej změnit. Dalším důležitým bodem je důkladné otestování aplikace. Aplikace je zatím testována s malým světem a malým rozsahem dat.

10 Reference

[1] ṄET *msdn* [online]. [cit. 2012-04-22].
Dostupné z: `http://msdn.microsoft.com/en-us/netframework/`

[2] ASP.NET *msdn* [online]. [cit. 2012-04-22].
Dostupné z: `http://msdn.microsoft.com/en-us/library/aa286485.aspx`

[3] SQL Server *msdn* [online]. [cit. 2012-04-22].
Dostupné z: `http://msdn.microsoft.com/en-us/sqlserver/`

[4] Úvod do LINQ *msdn* [online]. [cit. 2012-04-22].
Dostupné z: `http://msdn.microsoft.com/en-us/library/bb397926.aspx`

[5] Pirkl, Josef, *Řešené příklady v C#* české Budějovice: Protisk s.r.o., 2005.

A CD

Přiložené CD obsahuje:

- zdrojové kódy aplikac
- zkompilovanou aplikaci.
- knihovnu UserBaseGame pro hráče
- SQL skripty pro vytvoření databáze a entit
- SQL skripty a program pro vytvoření ukázkového světa
- ukázkové programy robotů pro ukázkový svět včetně již zkompilovaných knihoven

B Datové slovníky

Název	Typ	Klíč	null	Popis
equipmentID	int	pk	ne	jednoznačná identifikace konkrétního vybavení
slotType	int	ne	ne	typ slotu pro dané vybavení
name	varchar(100)	ne	ne	textový popis vybavení
equipment_type	int	ne	ne	typ vybavení
time_to_work	int	ne	ne	časová prodleva při používání vybavení
price	int	ne	ne	standadní cena vybavení
item_type	int	ne	ano	typ předmětu se kterým pracuje
power_MovementDelay	int	ne	ano	čas nutný k posunu o 1 pole (pouze u pohonů)
power_TurningDelay	int	ne	ano	čas nutný k otočení o 90 stupňů(pouze u pohonů)
armor_reduction	int	ne	ano	pohlcení pancíře (pouze u pancíře)
armor_hp_bonus	int	ne	ano	bonusové zásahové body pancíře (pouze u pancíře)
weapon_range_max	int	ne	ano	maximální dosah (pouze u zbraní)
weapon_range_min	int	ne	ano	minimální dosah (pouze u zbraní)
weapon_damage	int	ne	ano	poškození (pouze u zbraní)
inventory_size	int	ne	ano	velikost nákladového prostoru (pouze u nákladového prostoru)
transmitter_range	int	ne	ano	dosah komunikátoru (pouze u komunikátorů)
radar_range	int	ne	ano	dosah radaru (pouze u radarů)
memory_unit_max_data	int	ne	ano	maximální počet záznamů v paměťové jednotce

Tabulka 1: Datový slovník DBEquipments

Název	Typ	Klíč	null	Popis
slotID	int	pk	ne	slot daného vybavení
robotID	int	pk fk	ne	ID robota, který dané vybavení používá(cizí klíč na DBRobot
equipID	int	ne	fk	konkretni typ vybaveni z DBEquipment
variables	varchar(1000)	ne	ne	pomocne promenne daneho vybaveni

Tabulka 2: Datový slovník DBRobotEquip

Název	Typ	Klíč	null	Popis
RID	int	pk	ne	jednoznačná identifikace robota
playerID	int	fk	ano	hráč který robota vlastní
robotArchitectureID	int	fk	ne	architektura robota
hitPoints	int	ne	ne	aktuální počet zásahových bodů robota
x	int	ne	ne	souřadnice x robota
y	int	ne	ne	souřadnice y robota
z	int	ne	ne	souradnice z robota (patro)
direction	int	ne	ne	směr natočení robota
private_key	int	ne	ne	klíč nutný k přístupu do veřejné paměťové jednotky
robot_state	varchar(100)	ne	ne	stav robota (zaptnutý či vypnutý)

Tabulka 3: Datový slovník DBRobot

Název	Typ	Klíč	null	Popis
architectureID	int	pk	ne	jednoznačná identifikace konkrétní architektury
S1	int	ne	ne	typ slotu 1
S2	int	ne	ne	typ slotu 2
S3	int	ne	ne	typ slotu 3
S4	int	ne	ne	typ slotu 4
S5	int	ne	ne	typ slotu 5
S6	int	ne	ano	typ slotu 6
S7	int	ne	ano	typ slotu 7
S8	int	ne	ano	typ slotu 8
S9	int	ne	ano	typ slotu 9
S10	int	ne	ano	typ slotu 10
maxHitPoints	int	ne	ne	maximalní počet zásahových bodů
maxFloor	int	ne	ano	maximalni patro
price_per_HP	int	ne	ano	cena za opravu jednoho zásahového bodu
price	int	ne	ano	cena architektury

Tabulka 4: Datový slovník DBRobotArchitecture

Název	Typ	Klíč	null	Popis
robot_ID	int	pk	ne	ID robota, vlastnící tato data
unit_type	int	pk	ne	typ jednotky(private nebo public)
value_number	int	ne	pk	index proměnné
value	int	ne	ne	hodnota proměnné

Tabulka 5: Datový slovník DBMemoryUnit

Název	Typ	Klíč	null	Popis
MessageID	int	pk	ne	identifikace zprávy
sender	int	ne	ne	identifikace odesílatele
recipient	int	ne	ne	identifikace příjemce
text	varchar(1000)	ne	ne	text
read	int	ne	ne	informace, zda byla zpráva přečtena

Tabulka 6: Datový slovník DBMessage

Název	Typ	Klíč	null	Popis
x	int	pk	ne	souřadnice X na mapě
y	int	pk	ne	souřadnice Y na mapě
floor	varchar(100)	pk	ne	patro mapy
map_cell_type	int	ne	ne	typ buňky mapy
item_type	int	ne	ano	podtyp buňky mapy
item_count	int	ne	ano	počet surovin na buňce
fixing_percent	int	ne	ano	zpevňění tunelu (pouze pro nezpěvněný tunel)
tower_ID	int	fk	ano	identifikace věže na buňce

Tabulka 7: Datový slovník DBMap

Název	Typ	Klíč	null	Popis
towerID	int	pk	ne	Identifikace věže
tower_type	int	ne	ne	typ věže
tower_price	float	ne	ano	modifikátor cen

Tabulka 8: Datový slovník DBTower

Název	Typ	Klíč	null	Popis
PID	int	pk	ne	jednoznačná identifikace hráče
nickName	varchar(50)	ne	ne	uživatelské jméno
password	varchar(50)	ne	ne	heslo hráče
email	varchar(100)	ne	ne	email hráče
TP	int	ne	ne	herní platidlo - stav

Tabulka 9: Datový slovník DBPlayer

Název	Typ	Klíč	null	Popis
tower_ID	int	pkfk	ne	identifikace věže
item_ID	int	pk	ne	typ předmětu
price	int	ne	ne	výkupní cena za kus

Tabulka 10: Datový slovník DBTowerPrices

Název	Typ	Klíč	null	Popis
tower_ID	int	pkfk	ne	identifikace věže
item_ID	int	pk	ne	typ vybavení
price	int	ne	ne	nákupní cena

Tabulka 11: Datový slovník DBMarket